1.

client = MongoClient('localhost'**, 27017**)  
  
dbname = config.dbname  
collectionname = config.showcodecn  
filespath = config.filespath  
extentions = [".jsp"**,** ".java"**,** ".css"**,** ".js"]

connection to MongoDB and file path, database name and collection name from config file.

2.

def get\_files(filespath**,** extentions): # function to return the file paths in the given directory with given extention  
 filelist = []  
 for r**,** d**,** f in walk(filespath):  
 #print(r, d, f)  
 for file in f:  
 for extention in extentions:  
 if file.endswith(extention) and file.count(".") == extention.count("."):  
 filepath = join(r**,** file)  
 filelist.append(filepath)  
 #print(filelist)  
 return filelist

This function is to iterate through all folders and fetch files ending with given extensions.

3.

def getExtensionType(file):  
 colname=config.componenttypecn  
 col = client[dbname][colname]  
 #file=file.split('\\')[-1]  
 component\_type=list(col.find({"component\_name" : file.split('\\')[-**1**]}**,**{'\_id':**0**}))  
 component\_type=component\_type[**0**]['component\_type']  
 return component\_type

This function is to fetch extension type from MongoDB for particular file.

4.

def show\_code(filespath):  
 *"""  
 this function is to fetch all the lines from java and jsp files and return a string with <br> tag attached after each line* ***:param*** *filespath: this is to call getfiles function to create list of java and jsp files* ***:return****: we are returning metadata, which contains component name as file name, component type as file extension, cosestring as string we are creating.  
 """* files = get\_files(filespath**,**extentions)  
 # print(file)  
 storage = []  
 js\_lines=[]  
 output = {}  
 METADATA = []  
 filename = ''  
 comment\_flag=False  
 for filename in files:  
 f = open(filename**,** 'r')  
 Lines = f.readlines()  
 if filename.strip().endswith(('.java'**,**'.js'**,**'.css')):  
  
 Lines = [line for line in Lines if line.strip()]  
 # for line in f.readlines():  
 # storage.append(line)  
 code\_string = '<br>'.join(Lines)  
 # print(code\_string)  
 output['component\_name'] = filename.split("\\")[-**1**]  
 output['component\_type'] = getExtensionType(filename)  
 output['codeString'] = code\_string  
  
 METADATA.append(copy.deepcopy(output))  
 output.clear()  
 code\_string=''  
 if filename.strip().endswith('jsp'):  
 # js\_lines.append('<!--')  
  
 for line in Lines:  
  
 if line.strip().\_\_contains\_\_('<!--'):  
 comment\_flag=True  
 if line.strip().\_\_contains\_\_('-->'):  
 comment\_flag=False  
 continue  
 if not comment\_flag:  
 line=line.replace('<'**,**''**,1**)  
 js\_lines.append(line)  
 # js\_lines.append('-->')  
 code\_string = '<br>'.join(js\_lines)  
 output['component\_name'] = filename.split("\\")[-**1**]  
 output['component\_type'] = getExtensionType(filename)  
 output['codeString'] = code\_string  
  
 METADATA.append(copy.deepcopy(output))  
 output.clear()  
 js\_lines.clear()  
 code\_string = ''  
  
  
  
 #print(json.dumps(METADATA,indent=4))  
 return METADATA

This function is to create json for each file with component name as file name and code string as total lines in a file with joined <br> tag after each line.

5.

def dbinsertfunction(dbname**,** collectionname):  
 *"""  
 this function is to update database by calling show code and getfiles functions* ***:param*** *dbname: database name from config file* ***:param*** *collectionname: collectionname from config file  
 """* output = show\_code(filespath)  
  
 # print(output)  
 col = client[dbname][collectionname]  
 if output != []:  
 if col.count\_documents({}) != **0**:  
 col.drop()  
 print("Deleted the old"**,** dbname**,** collectionname**,** "collection")  
  
 col.insert\_one({"type": "metadata"**,** "headers": [  
 ""  
 "component\_name"**,** "component\_type"**,** "codeString"  
  
 ]})  
 col.insert\_many(output)  
 print("Inserted the list of jsons of"**,** dbname**,** collectionname)  
  
 else:  
 print("There are no jsons in the output to insert in the DB"**,** dbname**,** collectionname)

This function is to insert output into MongoDB after checking whether the output is empty or not. If empty it returns not available, else it updates output to database.

6.

if \_\_name\_\_ == '\_\_main\_\_':  
 output = show\_code(filespath)  
 dbinsertfunction(dbname**,** collectionname)  
 if not os.path.exists("outputs//"):  
 os.makedirs("outputs//")  
 json.dump(output**,** open('outputs\\show\_code.json'**,** 'w')**,** indent=**4**)  
 pd.DataFrame(output).to\_excel("outputs\\show\_code.xlsx"**,** index=False)

This code is to run required functions and create json file, excel file for output.